Open Liquidity Marketplace(OLM)

Introduction

Blockchains offer a shared trust platform for application developers to create smart contracts that run autonomously and can compose together. One of the most prominent use cases of blockchains today is the creation of tokens and on-chain markets like DEXs to swap them.

Moreover, the rise of L1s and L2s has meant tokens and markets are distributed across networks. This has given rise to another primitive- bridges, that connect these networks and facilitate token transfers between them. With the growing number of tokens and chains, the boundaries between swaps & bridging are quickly disappearing.

Today’s bridging and swapping primitives come with varying trade-offs. Some DEXs focus on stable swaps, while others offer broader asset coverage. Certain bridges prioritize speed, often sacrificing security, while others cover fewer chains.

To provide a unified experience, aggregators emerged—integrating multiple primitives and routing users through the best option in real time, optimizing for price, speed, and reliability.

However, aggregators face a critical bottleneck: the pace at which new primitives and chains emerge far exceeds the pace at which integrations can be done. As ecosystems scale, aggregators struggle to keep up.

Meanwhile, the emergence of AI agents has made it easier to digest onchain data while offering personalized user execution paths. Yet they face a fundamental trust problem, users cannot blindly entrust agents with funds, as agents may misinterpret intent or hallucinate actions, leading to loss of assets.

This paper introduces the **Open Liquidity Marketplace (OLM)**—a novel design that enables *any* off-chain agent to discover and compete to execute user intents such as swaps, based on preferences like price, speed, or security.

Unlike closed aggregators, OLM creates an open, competitive environment that **incentivizes agents** to integrate new primitives and chains quickly, as doing so directly improves their ability to win user orders. At the same time, user funds remain secure through on-chain smart contracts and crypto economics.

Ultimately, we believe an efficient, open and scalable marketplace design is essential:

- For users, it enables confident, on-chain execution with maximum choice.
- For primitives, it delivers access to users via an open discovery layer.
- For the ecosystem, an open marketplaces drive innovation through competition.

Design Goals

Design goals for the OLM:

- Creating an open and neutral marketplace

- A unified system with the ability for orders to be filled by off-chain liquidity sources (market-makers) or be routed to onchain liquidity primitives through open competition
- Allowing users to customize their preferences for price, speed, or security
- Offer protection against reverted transactions and MEV extraction for users, as well as allow soft-cancels for off-chain agents

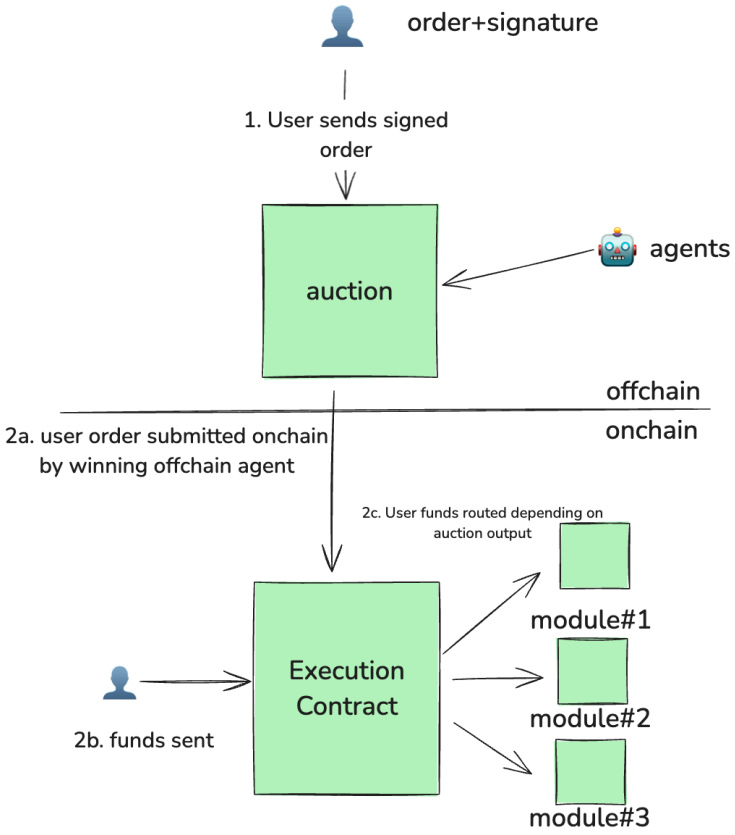
Mechanism — High Level Overview

The Open Liquidity Marketplace enables coordination between users and off-chain agents to execute liquidity-based actions like swaps, depositing tokens to smart-contracts or minting NFTs across networks.

It leverages a combination of crypto-economics, cryptographic proofs and smart-contracts to make sure that any off-chain agent can fulfill user requests, while ensuring users don't need to trust agents with their funds.

At a high level, order fulfillment via OLM works in 3 major steps:

- **User signs order**: User defines the inputs (token & chain), outputs (token & chain), along with preferences for price, latency or security
- **Auction**: User order is bid upon by off-chain agents. The auction matches the order with the agent maximizing user preferences
- **Order Execution:** The winning agent gets authorization to execute the user order on-chain. This is done via the execution contract and 3 modules with different functions



Marketplace Participants and Modules

OLM orders can be fulfilled in 3 ways by different types of off-chain agents, creating a competitive and highly liquid marketplace:

- **Escrow system**: Fillers like market-makers use their active inventory on-chain to fulfill user orders
- **Staking system**: Solvers who specialize in routing user funds to the most efficient venues, requiring only passively staked collateral
- **Proof system:** Provers who leverage OLM's native proving systems to route user funds by relaying cryptographic proofs

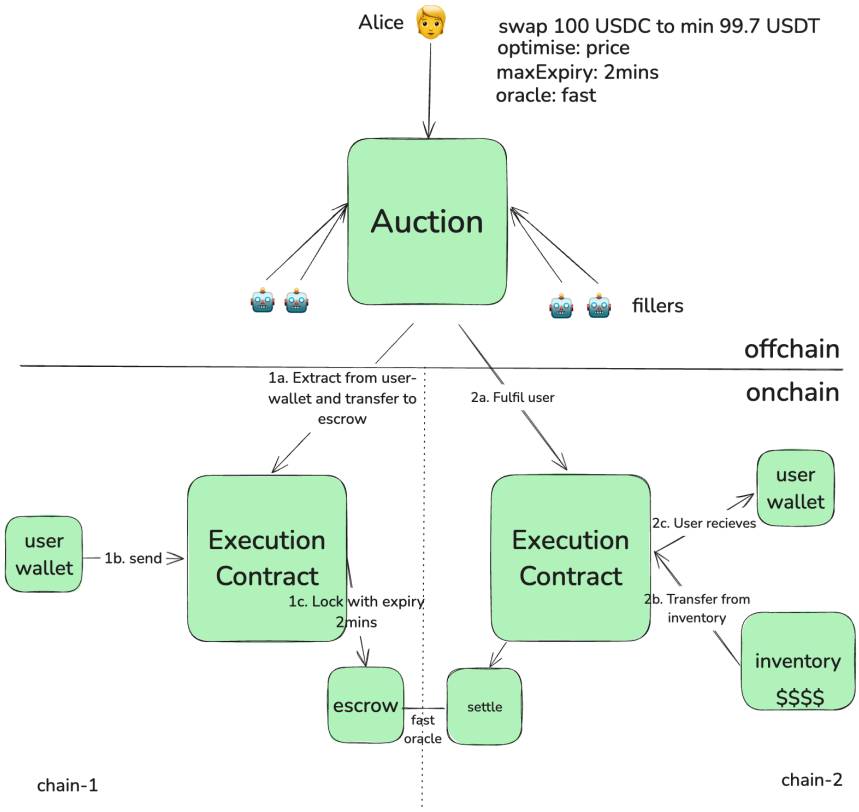
These are the 3 modules that agents call while submitting the user order to the execution contract. Let’s walk through the system and the modules in more detail.

Escrow-module

This execution is done by Fillers, agents with capability to execute sophisticated on-chain operations using their own capital inventory to fulfill orders across chains.

Transaction Flow

1. Alice sends a signed order stating inputs (chain/s, tokens), desired minimum output (chain/s, tokens) along with her preferences on price (minOutput), latency (via param maxExpiry), security (oracleID) to an auction
2. Off-chain agents of the `filler` type that hold inventory (USDC, USDT etc.) on-chain can bid on this order to win its execution rights
3. The winning Filler interacts with an onchain smart-contract and is able to pull funds from the user (using the signature from the first step) and send it to an escrow-system where funds remain locked until either of the following is true:
    a. maxExpiry is reached, so user can claim funds back from the escrow-system
    b. The selected oracle claims user received output
4. In the happy case, Alice's order gets fulfilled on the target chain by transferring necessary capital using the agent's inventory
5. The Filler can then trigger settlement through the user-selected oracle, allowing them to reclaim their funds from the escrow



OLM acts as a lightweight marketplace, allowing users and Fillers to match with each other while maintaining full control over execution, including the choice of settlement oracle. For example, in interactions between two chains within the Superchain, users can choose OP-Interop, AggLayer, or any other oracle system for settlement.

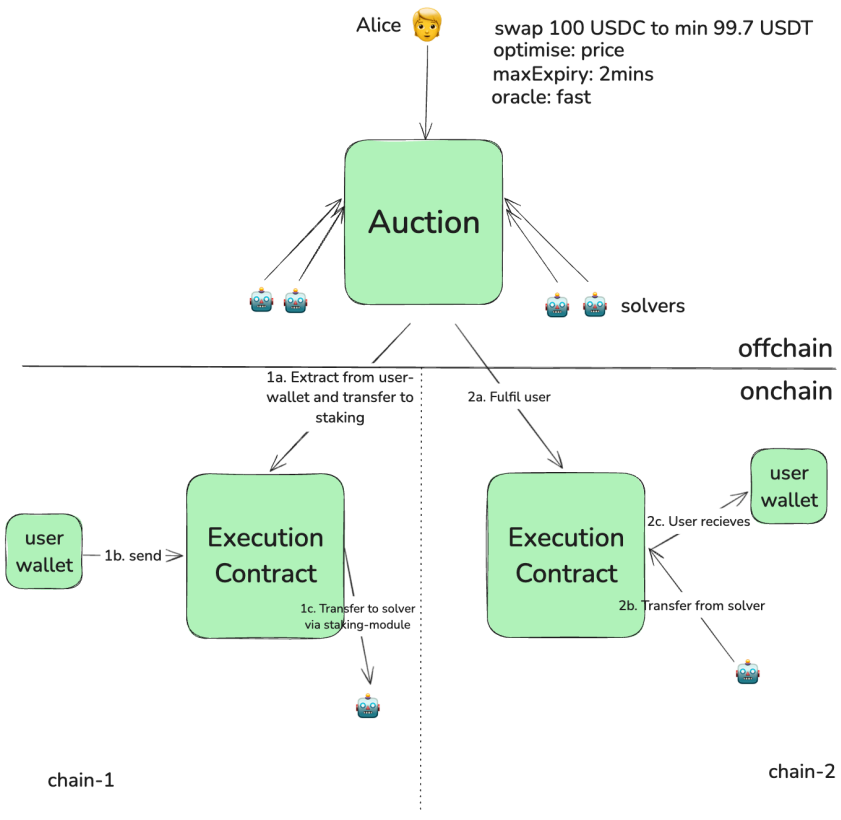
Staking-module

This execution is done by Solvers, specialised entities capable of performing optimal routing across integrated liquidity sources. The solver have staked assets but doesn't need any active capital.

Transaction Flow

1. Alice sends a signed order stating, inputs (chain/s, tokens), desired minimum output (chain/s, tokens) along with her preferences on price (minOutput), latency (via param maxExpiry), security (oracleID) to an auction
2. Off-chain agents of the `solver` type that stake assets (Aave/Morpho receipt tokens or native assets like ETH/USDC) will be able to bid on this order

3. The winning Solver interacts with an onchain smart-contract and is able to pull funds from Alice (using the signature from step 1) and send it to the staking-system
    a. The staking system also receives attestation from the oracle user selected to ensure solver has enough staked
    b. The staking system allows users to claim funds from the Solver's stake if they haven't been filled by maxExpiry time using the oracle attestation
4. Solvers get funds from Alice and are free to route them to the destination however they want. For ex: using bridges or DEXs
    a. They can also fill using their own inventory like a Filler would. However, in this case, they don't need to perform any settlement as they have already received funds from Alice on the input network



OLM incentivizes Solvers to integrate the latest primitives (e.g., DEXs, bridges). Solvers can route through mint/burn bridges or utilize CEX deposit/withdraw flows

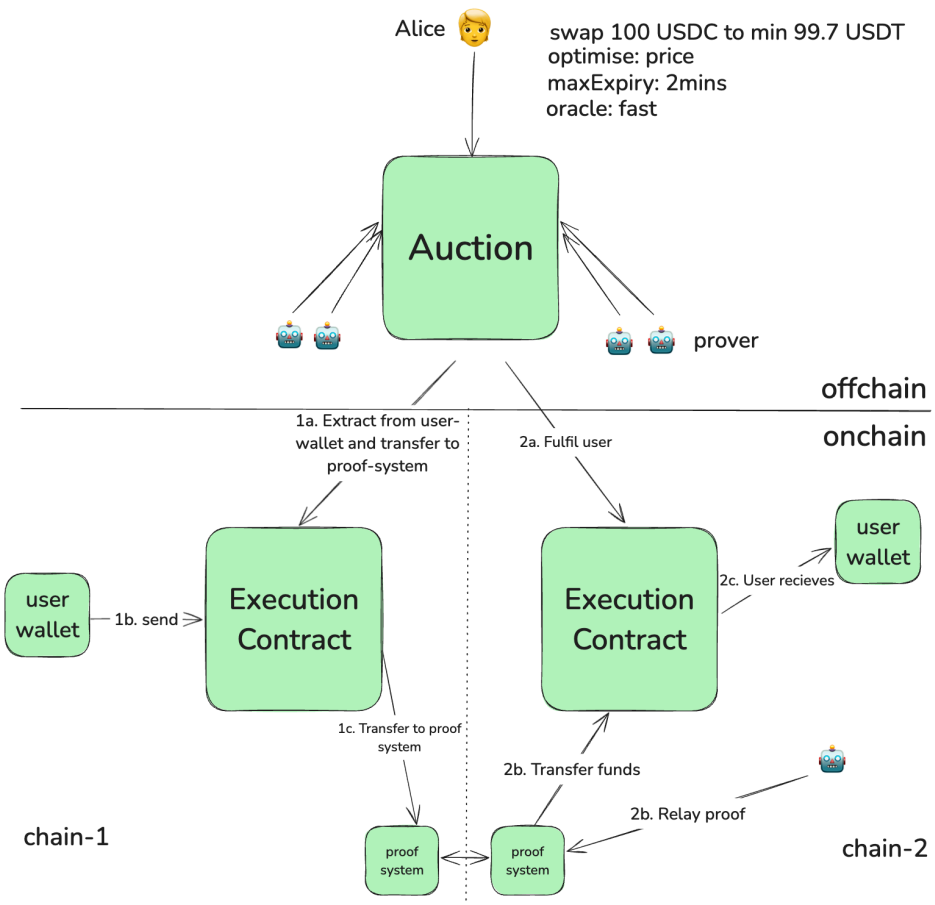
to execute user orders, improving efficiency and unlocking deeper liquidity. Any capital required by Solvers can come from re-staked assets leading to increase capital efficiency.

Proof-module

Provers are entities that understand how to create proofs for various integrated proof-systems like Circle, OP Interop or AggLayer within OLM. Proof systems are whitelisted & ingested into the OLM due to high guarantees of reliability & trust. They are integrated into OLM and are considered part of the protocol to allow anyone to access these systems without needing to stake.

Transaction Flow

1. Alice sends a signed order stating, inputs (chain/s, tokens), desired minimum output (chain/s, tokens) along with her preferences on latency (via param maxExpiry), security (oracleID) to an auction
2. Off-chain agents of the `prover` type can access one of the whitelisted proof-systems to fulfil user requests and hence bid on the user order
    a. Proof-systems are ingested into OLM due to the high guarantees of reliability/trust they provide
    b. Proof-systems don't take any inputs from Provers and operate maximally on-chain
3. The winning Prover interacts with an on-chain smart-contract and is able to pull funds from the user (using the signature from the first step) and send it to the proof-system
    a. The proof-system is able to burn tokens on that network
    b. The Prover is then able to relay the proof-of-burn on the output network and forwards Alice those funds
    c. The Prover gets paid after the user receives the funds. In case the Prover fails to relay, Alice can do so herself



Proof-systems on the OLM are permissioned and are expected to be added via governance or other mechanisms. Proof-systems are trusted to be reliable, secure and maximally on-chain such that anyone can generate these proofs. Provers don't need any capital to perform within OLM as they access the integrated proof-systems.

Extensions

Any-2-Any Swaps

When users want to swap a token for another token across different networks, like swap Aave@Polygon to Brett@Base, currently there are only 2 ways to go about it:

- **Multi-Step**: User first completes a swap + bridge from Polygon to Base, swapping Aave to USDC on Polygon & then bridging it over to Base. Fresh calldata for the swap is prepared and executed on Base. This requires the user to manually perform this multistep interaction which is horrible UX
- **Execute via swap-call data:** Many bridges enable executing calldata post bridge completion, which can be leveraged to perform the swap. However, since swaps are latency sensitive, users face failing swaps after bridging or need to set high-slippage in the calldata. This ends up leaking significant value to MEV bots
    - For example, consider this transaction by Sushiswap and Stargate where the user's swap was backrun by this transaction

To summarize, current approaches come with tradeoffs:

- Perform a painful multi-step process across networks which is terrible UX
- Leak a significant amount of of value to MEV, also terrible UX

However, with OLM we can avoid these completely. Off-chain agents perform the multi-step on-chain interactions on behalf of the users. They are able to leverage realtime data and prepare fresh swap-data at runtime instead of relying on static swap-data passed by bridges.

This removes the burden of performing these processes for them while avoiding any value leakage to MEV, making OLM particularly optimized for swaps between networks.

Same-Chain Swaps

The same design can also support same-chain swaps, where users receive output tokens on the same network as their input tokens. The design principle to facilitate these swaps are still intact: an open liquidity market that can leverage off-chain liquidity as well as route to on-chain primitives - like DEXs - efficiently.

Execution for same-chain swaps is simpler because atomicity is guaranteed, unlike when dealing with different chains. A proof-system does not need to exist in this model, as the trust assumption is simply the network itself.

Oracle Handshake

OLM is designed to give users maximum choice and control. When placing an order, users can define their own trust preferences as described above, using oracleID. Off-chain agents that meet these preferences can fulfill the order.

This lets users navigate the trust vs. efficiency spectrum: for a $10 swap, they might prioritize speed and low cost; for a $1M swap, they might opt for more secure settlement.

MEV Redistribution & Soft-Cancels

Whenever protocols coordinate with off-chain agents via on-chain auctions, it leads to wasted gas and leaking value to validators due to reverted transactions. While there are some networks that provide revert-protection at the chain level, we think it is best handled at the application layer via auctions run offchain.

When a user sends an intent to an intent-bridge like, for example, Across, Fillers on the target network race to fill it, as the one who fills first is successful at keeping the profit.

Say the order has a fee of $10 and some MEV value of $2 due to some calldata execution as part of the intent. Fillers have a margin of $12 to bid up to at most get into the block early. Fillers compete against each other, progressively paying more to validators to get into the block early. Say the successful filler was able to get in by paying 10$: this is now value that went to validators/sequencers that the OLM could otherwise redistribute back to users.

Moreover, the economics of reverts affect participation by off-chain agents in these auctions. If you post a bid that ultimately doesn't win, you don’t just lose the

fees from order, you also lose money on the revert. Below is a screenshot showing 1.7M failed transactions for Across by fillers trying to fulfil user-orders. OLM allows fillers to have soft cancels and no reverts.

Query results New Query

failure_rate_pct	total_failed_txns
21.5	1717282

Snapshot of failed transactions by fillers recorded in Feb 2025 across Base, OP, Arbitrum, Polygon and Ethereum on Across Bridge

Gas Abstraction & Revert Protection

With OLM, users benefit from a chain-abstracted experience: they no longer need to interact directly with blockchains. Instead, they delegate execution in a controlled way to off-chain agents who fulfill their orders on-chain.

This architecture provides two huge UX benefits: gas abstraction and revert protection.

Gas abstraction: Allows users to pay network fees using the tokens they're swapping from, rather than needing to hold a blockchain’s native token to cover gas. For example, if you're swapping USDC to USDT, you can pay fees in USDC instead of ETH.

Revert protection: with OLM, users don't pay for failed transactions. This is especially valuable in environments with fast block times where rapid price movement can cause on-chain swaps to fail. Off-chain agents absorb this risk, making the experience smoother and more reliable for users.

Pre-Confirmations for Off-Chain Agents

Swapping assets between networks has inherent latency since the transactions happen on both the chains sequentially. Many innovative designs like OneBalance, Magic-Spend++, and The-Compact are making this faster by introducing an off-chain engine.

They receive orders from users and issue pre-confirmations to agents such that they can process the order on the target network without waiting for the action to happen on the source network.

Since OLM sits between users and off-chain agents to perform matching between them, it is uniquely positioned to offer pre-confirmations to off-chain agents as part of the matching.

This unlocks zero-latency cross-network swaps for users, dramatically improving UX without compromising on safety.

Prior Work

Cross-Chain Intent Protocols

Intent bridges like Across, Debridge and several others are great examples of using off-chain agents to fulfil user orders. These protocols rely on fillers or market-makers to fulfil orders. Due to the high degree of sophistication required to become a Filler, the competitiveness and liquidity-profile of these systems is sub-optimal.

Secondly, due to their FIFO auction type, these protocols optimise for fast fulfilment over best execution or ability for users to have choice over trust-level.

OLM is able to leverage not just Fillers but also Solvers and Provers to fulfil the user orders. This allows OLM to sidestep liquidity constraints, making the marketplace competitive and able to offer the full spectrum of tradeoffs to users and integrators.

Intent-bridges are still very valuable as they optimise for speed and hence some users' orders might be routed via these primitives as part of OLM.

Aggregation Services

Aggregation services like BungeeV1 and LiFi/Jumper integrate multiple bridges and DEXs, offering users all of these integrations under a single interface and acting as a discovery platform for users.

However, since all integrations are done manually, the aggregation service is never able to catch up to all available integrations. This is further amplified by the exponential increase in number of networks.

Integrating asynchronous services like bridges, which offer no on-chain guarantees of reliability, introduces fragility into the system. As more integrations are added, the risk continues to grow: one integration breaking could affect the entire service for all users. This inability to scale undermines the usefulness of the service to its users.

OLM is built to solve these exact pain-points and some off-chain agents might leverage aggregation-services to route user orders wherever relevant.

CoW Swap

CoW Swap is a meta-aggregator targeted towards same-chain swaps and was indeed a big inspiration behind the OLM model. While CoW Swap enables same-chain swaps, OLM extends the model to deliver swaps between networks, leveraging different kinds of off-chain actors to make the marketplaces as competitive as possible.